Linux - Introduction aux bases
Utilisation en ligne de commande

Guillaume Allègre
Guillaume.Allegre@silecs.info

INP Formation Continue / CNFPT

2012

Licence Ceative Commons By - SA

Contribuer - Réutiliser

Ce document est rédigé en LaTeX+ Beamer.

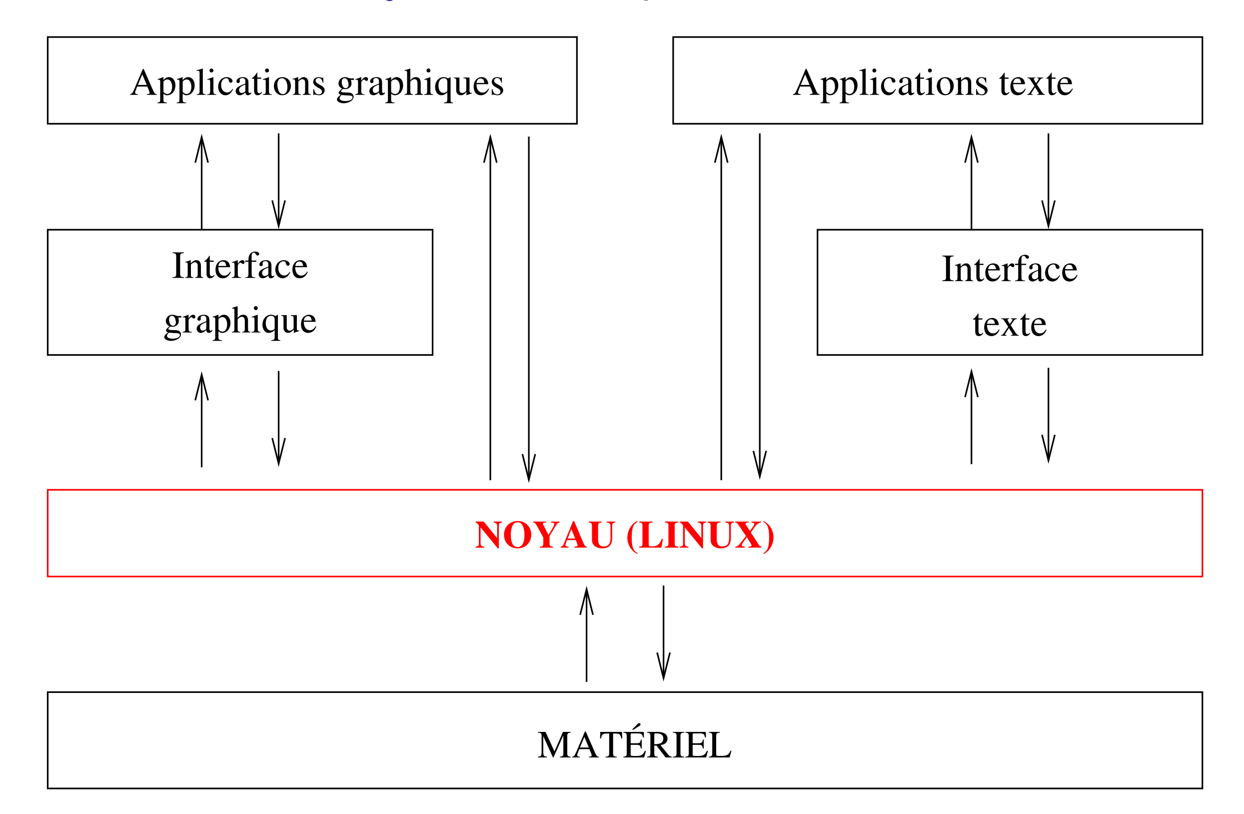
Conformément à la philosophie du logiciel libre, vous êtes encouragés à réutiliser, reproduire et modifier ce document, sous les conditions de la licence *Creative Commons, Attribution, Share alike 3.0* précédemment décrite.

J'accepte volontiers les remarques, suggestions d'améliorations, corrections et contributions à ce document

Vous pouvez obtenir les sources de ce document en m'écrivant, ou bien accéder au dépôt Mercurial des sources LaTeX :
`http://hg.silecs.info/hg/public/formations/linux/`
où vous pouvez naviguer (*browse*) ou télécharger une archive (zip, tgz ou tar.bz2).

Qu'est-ce que Linux ?

Architecture d'un système d'exploitation



Une histoire de famille : Unix

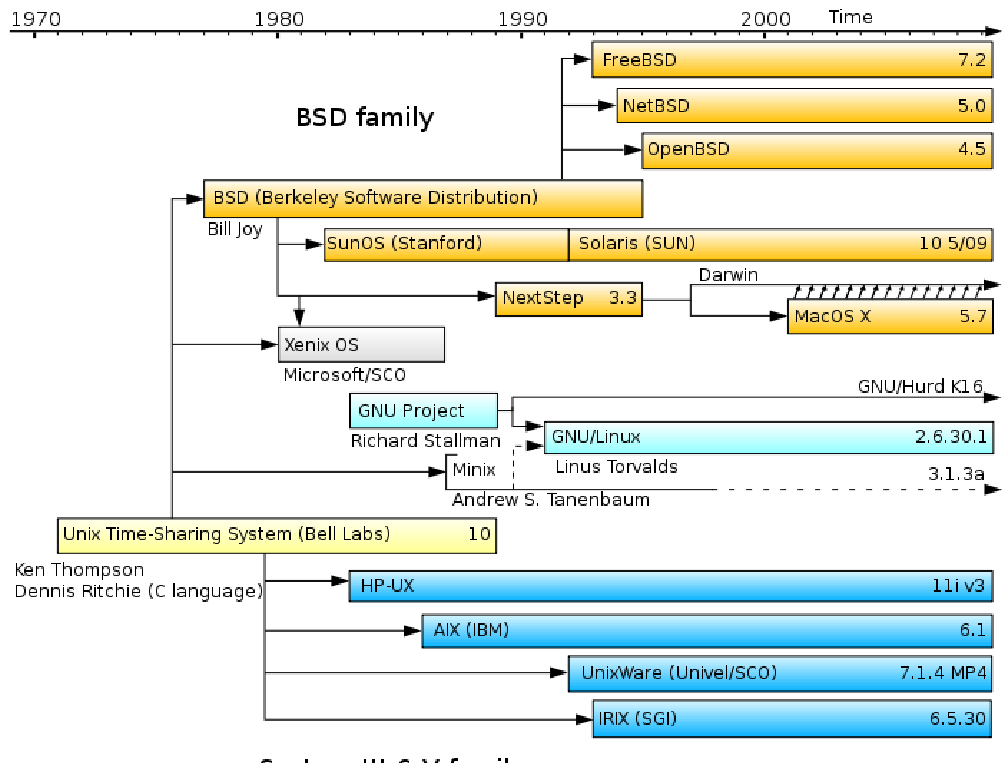
UNIX en quelques points :

1. apparu en 1969 à AT&T - Bell Labs., K. Thompson, D. Ritchie
2. beaucoup de dérivés : Solaris, AIX, BSD, OS X. . .
3. conçu comme un système professionnel :
   - orienté réseau,
   - multi-tâches,
   - multi-utilisateurs.
4. trois survivants propriétaires : Solaris (Sun), AIX (IBM), HP-UX

Une normalisation : POSIX (IEEE 1003) 1985-1998

1. 17 thèmes : Core, Real-time, Threads, Shell...
2. évolutions : POSIX :2001, POSIX :2004, POSIX :2008

Une brève histoire d'Unix

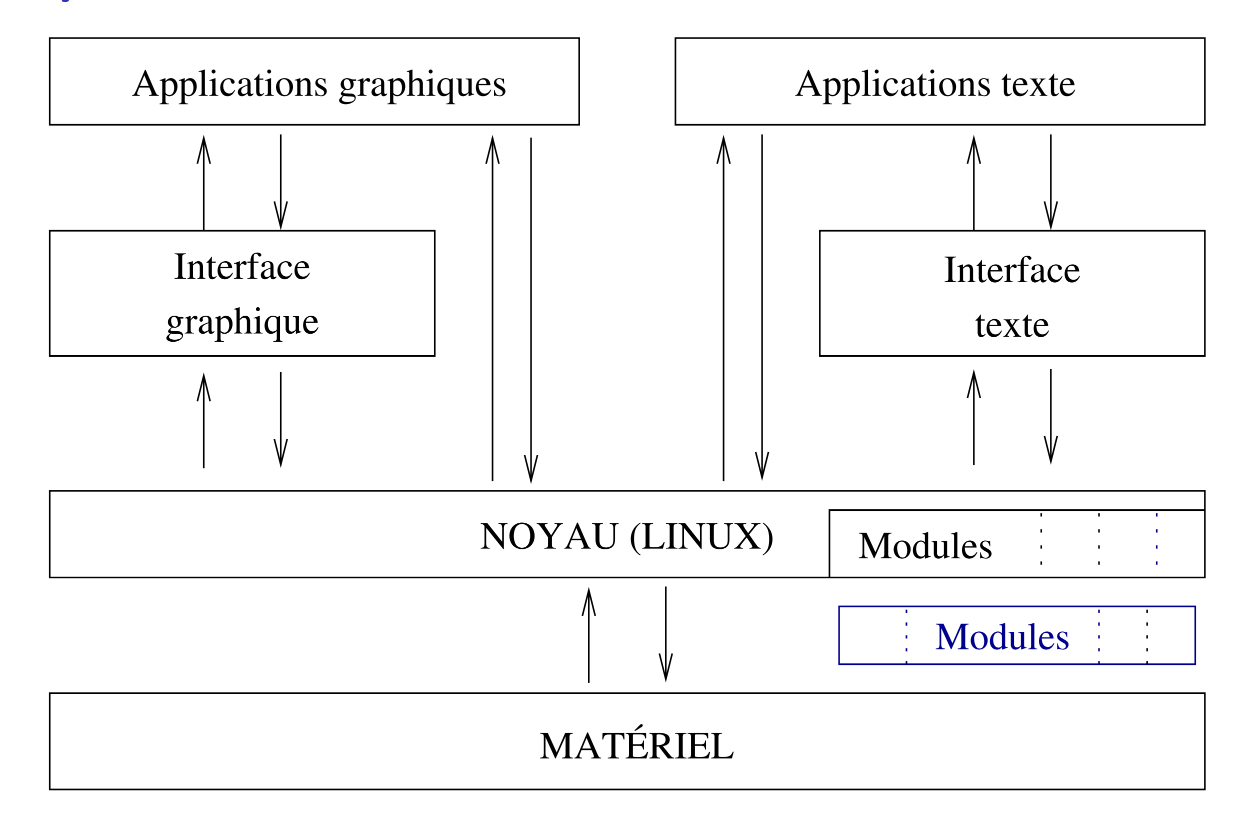


Domaine Public - Wikimedia Commons - Unix history.en.svg

Les spécificités de Linux

- créé en 1991 par *Linus Torvalds*, étudiant finlandais.
- logiciel libre
  - inscrit dans la mouvance GNU
  - sous licence GPL depuis 1992
  - fer de lance du logiciel libre
- développement décentralisé et collaboratif
- modulaire : chargement d'extension du noyau à la demande (pilotes. . .)
- portable : compatible avec un très grand nombre d'architectures.

Le système Linux



Principales différences GNU/Linux / Windows

1. Un ensemble très modulaire vs. un bloc monolithique
2. Une seule arborescence (*tout est fichier*)
3. Fichiers de configuration et éditeurs de texte
(pas de base de registres)
4. Importance de la ligne de commande (*une tâche, un outil*)
5. Profondément **réseau** et **multi-utilisateurs**

Linux et le libre

- Linux est un système d'exploitation sous **licence libre**
  1. liberté d'usage, sans restriction
  2. liberté d'étude du logiciel et de modification
  3. liberté de copie et diffusion
  4. liberté de diffusion des modifications

- Pour 2. : importance du **code source**
- Sphère privée (1-2) / sphère publique (3-4)

- Licence **GPLv2** : General Public License
  Il existe d'autres licences libres (ex : BSD, MPL...)

- Projet GNU : Le complément du noyau...

Le projet GNU : *GNU's Not Unix*

- Origine (1983) : réimplémentation libre des utilitaires Unix
  - `glibc` + `gcc` : GNU C library + GNU C Compiler
  - `binutils` (ld, as, gprof, nm, ar, strings...), make, gdb...
  - `coreutils` (ls, chmod, sort, du, nice...), grep, sed, awk
  - `bash` : shell compatible sh

- Récemment : focalisation sur les projets "stratégiques"
  - GNU Hurd : noyau libre (pas opérationnel, cf. Linux)
  - Gnu Privacy Guard : crypto personnelle (alternative à PGP)
  - Gnome : environnement de bureau (alternative à KDE)
  - Gnash : lecteur Flash libre (alternative à Adobe...)
  - ...

- Logiciels indépendants
  - Emacs (1976-) : éditeur texte original, alternatif à `vi`
  - GIMP : retouche d'images
  - Dia : conception de diagrammes
  - ...

Linux et le libre

- Il existe des logiciels propriétaires pour Linux (ex. serveur Oracle)
- Il existe des logiciels libres pour Windows... (ex. Apache, Mozilla Firefox, OpenOffice.org)
- Il existe d'autres OS libres (ex. FreeBSD)
- Libre n'est pas gratuit
  - parfois si : Linux est libre **et** gratuit
  - *freeware* : gratuit, pas libre (code source)
  - développements à façon : libre, pas gratuit...

Les distributions Linux

Leur rôle :

- Simplifier la vie de l'administrateur.

Une distribution comprend :

- le noyau Linux
- un système d'installation
- des logiciels applicatifs
- des outils d'administration
- Éventuellement
  - un support physique (boîte, CDROM, documentation. . .)
  - des services (maintenance, hotline, formation. . .)

Les distributions Linux - Diversité (2)

Près de 400 distributions actives.
Cf. http://distrowatch.com/ et http://futurist.se/gldt/

Causes de diversité :

1. Modèle de développement
   - communautaire : Slackware, Debian et certaines dérivées...
   - commerciale : la plupart des autres
2. Modèle d'administration
   - Installation des logiciels (.deb / .rpm / .tar.gz)
   - Services (Redhat / Fedora)
3. Spécialisation
   - Autonome : Knoppix, Kaella
   - Grand public : Ubuntu
   - Sécurité réseau : IP Cop
   - Localisation : Mandriva
   - Dépannage : System Rescue
   - Recompilation (performances) : Gentoo

Les distributions : la famille RedHat

- RedHat Linux (ancien modèle) : RH 1.0 (1994) à RH 9 (2003)
  - mise au point du format RPM (RedHat Package Manager)
- RedHat Enterprise Linux (RHEL) : depuis RHEL 3 (2003)
  - dernière : RHEL 6.2 (déc. 2011)
  - plusieurs variantes : Desktop, Workstation, ES, AS...
- Fedora (Core)
  - version communauté
  - dével. rapide (env. 2/an) depuis FC 1 (nov. 2003)
  - dernière : Fedora 16 (nov. 2011)
- CentOS
  - clone de RHEL, sans le service
  - utilise les sources fournies par RedHat
- autres utilisatrices de RPM : Mandriva, Novell SuSE...

Les distributions : la famille Debian

- Debian GNU/Linux : 1.0 (1996) à 6.0 Squeeze (fév. 2011)
  - collaborative et non commerciale
  - essentiellement libre
  - format de paquets (avancé) .deb
  - dépôts et installation réseau
  - mises à jour régulières (6.0.4 jan. 2012)
- Ubuntu : commerciale (Canonical LTD, GBM)
  - installation simplifiée
  - deux sorties par an (ex. 11.04 et 11.10)
  - partiellement compatible Debian
  - basée sur Gnome, choix restreint de paquets
- Knoppix : distribution autonome (*live*)
  - s'exécute sans installation (depuis le CD et la RAM)
  - peut s'installer et se transformer en Debian

Cont Base Fichiers Outils Réseau Scripts Installation Linux ? Le libre Distros Économie...

Administration Linux : les paquets

Chaque distribution propose un système d'installation de logiciels via des paquets (.deb / .rpm / .tar.gz).

Avantages :

- Normalisation
- Simplification
- Gestion des dépendances
- Mise à jour centralisée

Remarque : possible d'installer un programme sans ce procédé.

Un effort de normalisation pour Linux

- Linux Standard Base (LSB)
  - 2001 (1.0) - 2011 (4.1) ...
  - dérivée / inspirée de POSIX
  - indépendante des distributions (mais RPM-centrée)
  - normalisation des composants (bibliothèques...)
  - normalisation de la hiérarchie (FHS)
  - fourniture de tests de compatibilité
- Linux Foundation
  - créée en 2007 : fusion de l'OSDL et du FSG
  - sponsorise Linus Torvalds et d'autres développeurs
  - édite la LSB et d'autres documents de référence (OpenPrinting...)

Les communautés du libre...

- Notion de communauté
  - modèle propriétaire : césure développeurs / utilisateurs
  - modèle libre : tous les intermédiaires
- Participation à la communauté
  - le « pot commun » : mutualisation et réciprocité
  - support informel (forums, listes de diffusion)
  - rapports de bugs (et plus)
- Émergence d'outils techniques
  - Internet et communication (mail, newsgroups)
  - Gestionnaires de versions (code source)
  - Suivi de bugs / de tickets (Bugzilla...)
  - SourceForge, GForge. . .

Logiciel libre : économie de services

- Économie de l'immatériel
  - Une idée n'est pas un bien matériel
  - Le partage n'appauvrit pas
  - Le logiciel "en boîte" est un leurre
- Des modèles économiques multiples
  - Constructeur : vend du matériel, donne le logiciel
  - Services : expertise, formation, développements sur mesure
  - Éditeur
    - hébergement (Software as a Service), *cloud*
    - audit, expertise
    - double licence, licence "chronodégradable"
- Quelques points délicats
  - Relations éditeur / communauté
  - Conditions de contribution
  - L'*open source* comme argument marketing

La "professionnalisation" de Linux

- Linux Foundation
- Linux Standard Base
- Linux Professional Institute : certification
  - niveau 1 : junior
  - niveau 2 : avancé
  - niveau 3 : senior (spécialisé)

Avantages du libre

- Éthique : collaboration, partage
  concerne : enseignement, administrations...
- Économie : redéploiement coûts achat vers services (formation, support)
- Pérennité et indépendance : moins lié à un éditeur
- Souplesse : adaptabilité aux besoins
- Mutualisation (coûts de développement)
  concerne : administration, collectivités locales...

Linux pour l'utilisateur

Linux au démarrage *101.2*

En général (poste de travail) :

1. BIOS / EFI. . .
2. Chargeur de démarrage (GRUB ou LILO)
3. Mode texte
4. Mode graphique
5. Authentification par login + mot de passe
6. Bureau utilisateur (KDE, Gnome, XFCE. . .)

On peut aussi avoir (serveur) :

1. BIOS / EFI . . .
2. Chargeur de démarrage (GRUB / LILO)
3. Mode texte
4. Authentification par login + mot de passe
5. Shell (en mode console)

Changement de mode : **Ctrl** + **Alt** + **F1-F6/F7**

Ligne de commande vs interface graphique

- Inconvénients de la ligne de commande
  - apprentissage plus long
  - efficacité moindre (utilisateur débutant)
  - mémorisation nécessaire (partiellement)
  - domaine d'application limité (mais pas tant que ça...)
- Avantages de la ligne de commande
  - automatisation aisée
  - efficacité (rapidité) supérieure (utilisateur aguerri)
  - ressources négligeables (CPU, réseau...)
  - expressivité plus forte (options)
  - modularité et extensibilité (une tâche, un outil)
  - compréhension et contrôle des actions

Session utilisateur

Comptes utilisateurs :

- session : login/mot de passe *(username/password)*
- homedir : répertoire personnel
- permissions d'accès aux ressources (fichiers, processus) :
  - utilisateur
  - groupe
  - autres

Un compte unique d'administrateur (super-utilisateur) : root

Des comptes "services"

- pour les tâches système : mail, impressions, ...
- des droits restreints (par rapport à root)
- sécurité accrue en cas de bug ou compromission

Découverte du shell - 1 *103.1*

Le prompt (invite de commandes)

- utilisateur courant
- nom de machine
- répertoire courant
- $ ou # : terminateur
- ... configurable à l'extrême
- un curseur !

Découverte du shell - quelques commandes *103.1*

id	Qui suis-je ?
pwd	Où suis-je ?
uname -a	À qui ai-je l'honneur ?
lsb_release -a	Mais encore ?

ls	Liste les fichiers
cd	Changement du répertoire courant
man	Page de manuel d'une commande
cat	Affiche le contenu d'un fichier

Commandes : syntaxe générale *103.1*

Syntaxe :

`commande [options] [--] [paramètres]`

Exemples :

- `ls --help`
- `ls -a`
- `ls --all`
- `ls -al`
- `ls -l .bashrc`
- `ls -w 60`
- `ls -w60`
- `ls --width=60`

Remarques : quelques exceptions

- `find . -name '*.tex' -print`
- `dd if=/dev/hda1 of=hda.img bs=512`

Commandes internes et externes *103.1*

- Commandes d'identification
  - `which` : commandes externes (fichiers)
  - `type (-a)` : commandes connues du shell
  - `whereis` : binaire et page de man d'une commande
- Les principaux types de commandes
  - **commande externe (fichier exécutable) - hashed**
  - **commande interne ou primitive shell (builtin)**
  - alias
  - fonction shell
  - mot-clé du shell, ex. `if`, `for`
- Exo : déterminer le type des commandes suivantes
  `cd, cp, ls, which, type, echo`

Documentation - formats et logiciels *103.1*

- aide en ligne de commande
  `ls --help`
- aide de bash : `help` (commandes internes)
- pages de manuel : `man`
  cf page suivante
- `info` : la documentation GNU (voir aussi `pinfo`, `tkinfo`)
- et encore : des pages .html, des fichiers `README`, `.chm`...
  voir `/usr/share/doc/`
- navigateurs d'aide (Gnome, KDE...) : interne, man, info...

Documentation - manpages

103.1

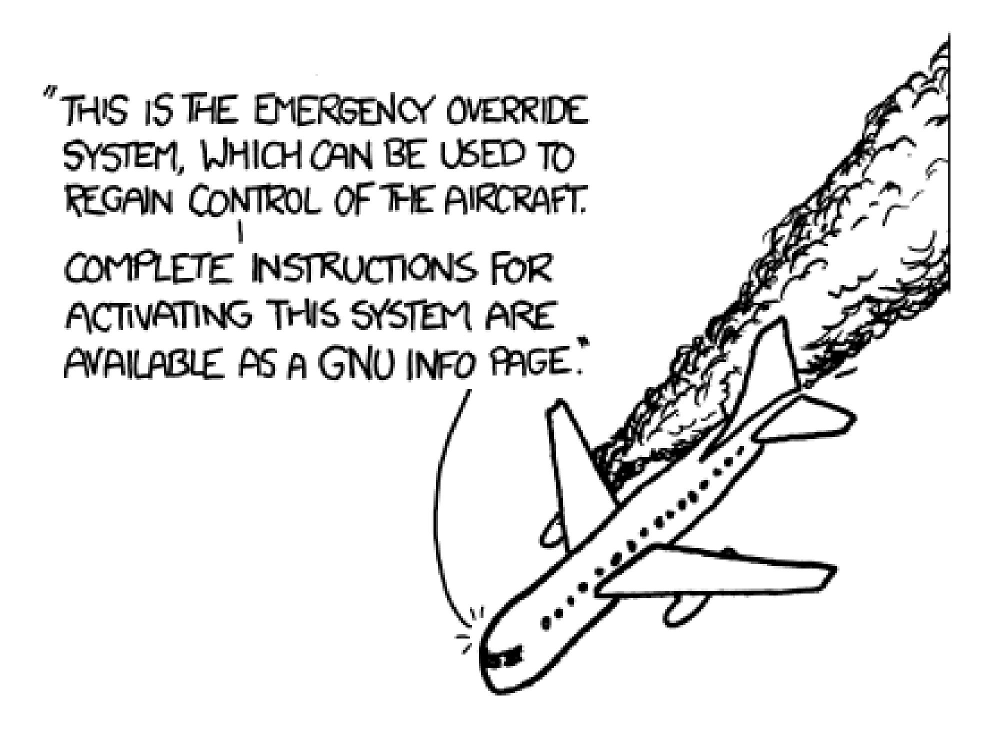
- man ls, man man
- Neuf sections

1. commandes util.	4. périphériques	7. “conventions”
2. appels noyau (C)	**5. fichiers conf.**	**8. commandes admin.**
3. appels bibli. (C)	6. jeux	9. routines noyau

  man (1) man, man 7 man
- Parties génériques : Nom, Synopsis, Description, Auteurs, Voir aussi...
- Pager less intégré : défilement
  - recherche : /motif, n, N, . . .
  - marqueurs : m a . . ., ’ a . . .
- survivant du système roff/nroff/groff (formatage à balises)

XKCD 912 - Manual Override

103.1



 http://xkcd.com/912/

Gestion des fichiers

Gestion des fichiers et répertoires *103.3*

- Commandes courantes
  - informatives : `ls, cat`
  - modificatrices : `touch, cp, mv, rm`
  - répertoires (informatives) : `pwd, cd, du, tree`
  - répertoires (modificatrices) : `mkdir, rmdir`
- Spécificités Unix
  - infos détaillées : `stat`
  - permissions : `chmod, chown,chgrp`
  - liens : `ln (-s), readlink`

Récapitulatif : chemins relatifs et absolus *103.3*

- Chemins absolus : exemples
  - `ls /home/stg1/Linux`
  - `ls ~stg1/Linux`
  - `ls ~/Linux`
- Chemins relatifs : exemples
  - `ls Linux`
  - `ls ./Linux`
  - `ls ../AutreRepertoire`
- Ne pas confondre : fichiers et répertoires cachés
  ex. `ls -l ./.bashrc`

Permissions sur les entrées de répertoires *103.3*

Trois cibles de permissions

- u=user : utilisateur propriétaire
- g=group : groupe propriétaire
- o=other : tous les autres
- (a=all : tout le monde)

Trois types de droits

	sur fichier	sur répertoire
r=read	lecture	listage
w=write	écriture	ajout/suppression fichier
x=exec	exécution	traversée
X=exec	conditionnelle	traversée

Permissions : Travaux pratiques *103.3*

Exercice : Remise des devoirs

Un prof cherche à récolter les programmes rédigés par ses étudiants dans un répertoire commun. Tous doivent pouvoir déposer un fichier, mais aucun ne doit pouvoir lire les autres fichiers déposés.

- Mettre en place la configuration nécessaire, ouverte à tous les utilisateurs.
- Comment restreindre le dépôt à un groupe de TP, nommé `tp01` ?
- Comment éviter les conflits de nommage entre plusieurs étudiants ?

Permissions Unix - Compléments *103.3*

Notation octale

- r=4, w=2, x=1
- ex. rwx r-x r-- = 754

SUID et SGID

- suid : changement d'UID à l'exécution `chmod u+s fichier`
- sgid : changement de GID `chmod g+s fichier`

Sticky bit

- fichier : obsolète
- répertoire : restriction à l'ajout/suppression d'entrées
  `chmod +t rép.`

- Extension ACL : Access Control List
- `man chmod`

Métadonnées Unix *103.3*

- Nom (entrée répertoire)
- Horodatage
  - `atime` (access) : date de dernier accès (lecture) `ls -lu`
  - `ctime` (change) : date de modification des métadonnées (inode) `ls -lc`
  - `mtime` (modification) : date de modification du contenu `ls -l`
  - Exo : que devient l'horodatage en cas de : `cat`, `vim` (avec et sans modif), `mv` (renommage), `chmod` ?
- Permissions
  - utilisateur propriétaire : `uid` numérique
  - groupe propriétaire : `gid` numérique
  - `mode` r,w,x. . . (champ de bits) ex. `0644/-rw-r-r-`
- Auxiliaires
  - type de fichier (régulier, répertoire. . .)
  - taille en octets
  - compteur de liens
  - pointeurs sur les blocs de contenu (. . .)

Liens physiques et liens symboliques - en pratique *103.3*

```
$ touch fichier
$ cp fichier fichier-cp
$ ln fichier fichier-ln      #lien physique
$ ln -s fichier fichier-lns  #lien symbolique

$ ls --inode --long

2080774 -rw-r--r-- 2  [...] fichier
2080775 -rw-r--r-- 1  [...] fichier-cp
2080774 -rw-r--r-- 2  [...] fichier-ln
2080776 lrwxrwxrwx 1  [...] fichier-lns -> fichier
```

Liens physiques et liens symboliques - inodes *103.3*

Usages

- Alternatives, ex. `vim -> /usr/bin/vim.basic`
- Rétro-compatibilité, ex. `/tmp -> /var/tmp`
- "Raccourcis"

Structure du système de fichiers - inodes

Liens physiques et liens symboliques - comparaison *103.3*

	lien symbolique	**lien dur**
pointe sur	entrée de répertoire	inode
rôle	asymétrique	symétrique
cible	tout type	fichier régulier
chemin cible	absolu ou relatif	N.A. (inode)
système de fichiers	interne ou externe	interne
cohérence	peut être cassé	jamais cassé
stockage	fichier (spécial)	entrée de répertoire

Globbing (expansion des noms de fichiers) *103.3*

But

Ne pas avoir à taper le nom de tous les fichiers en argument.

Exemple

```
ls *.rc
```

Caractères spéciaux

- `*` Tout
- `?` Un caractère quelconque
- [a-z] Un caractère parmi ceux listés

Protections contre l'interprétation par le shell

- "..." Protège partiellement ... de l'interprétation par le shell
- '...' Aucune interprétation de ...
- \... Aucune interprétation du caractère suivant

Pour aller plus loin : globbing personnalisé *103.3*

- Personnalisation du globbing
  - Commande shell `shopt (-s | -u) option`
  - Variable d'environnement : `$GLOBIGNORE`
- Options concernant le globbing

dotglob	inclut les fichiers "cachés"
failglob	erreur si rien ne correspond
globstar	récursif avec `**` et `**/`
nocaseglob	insensible à la casse
nullglob	chaîne vide si rien ne correspond
extglob	motifs étendus

Archives : tar... *103.3*

- L'archivage : rassembler plusieurs fichiers en un seul.
  `tar -c`, `tar -x`, `tar -t`
  `-f archive.tar` : spécifier le fichier archive (sinon flux)

- La compression
  - `gzip` + `gunzip` (ou `tar -z ...`)
  - `bzip2` + `bunzip2` (ou `tar -j ...`)

- Exercice
  1. Prendre connaissance du contenu de `tp-access.tgz`
  2. Décompresser l'archive
  3. Créer une archive compressée avec les 20 premiers fichiers
  4. Compresser individuellement les 20 derniers

- Voir aussi : `zcat`, `zless` ...
- Unix historique : `cpio` + `compress` (.Z)

Rechercher un fichier... 1/2 Indexation *104.7*

- `locate` : recherche rapide dans une base de données
  - `locate` (GNU) : source `findutils`
    - `-r` expression régulière, ex. `-r fst.b`
    - `-S` statistiques ...
  - `slocate` (obsolète) : + permissions
  - `mlocate` : + optimisation base

  - TP : Avec `updatedb` : lancer une indexation personnelle de son répertoire

- Fichiers et paquets (distribution)
  - (Debian) `dlocate` : recherche parmi les paquets installés
    alternative rapide à `dpkg -S`
  - (RH) `rpm -qf`

Rechercher un fichier - 2/2 Find *104.7*

- `find` : recherche multicritères
  `find /etc/ -size +10k -ctime -10 -printf '%s %p'`
  - répertoire de départ (/etc)
  - options de sélection (size, ctime)
  - options d'action (printf)

  - Toujours à jour
  - Potentiellement plus long que `locate`
- Exercices
  - Trouver le nombre d'entrées de répertoire de chaque type sous /, sans changer de système de fichiers (`-xdev`).

  - Pour les quatre types minoritaires, afficher les entrées.

Redirections - canaux *103.4*

Le shell définit 3 canaux

STDIN (0)	entrée standard - clavier par défaut
STDOUT (1)	sortie standard - écran (terminal) par défaut
STDERR (2)	sortie d'erreur - écran (terminal) par défaut

Redirection

`ls -1 > liste.txt`
La sortie du programme `ls` est **redirigée** vers un fichier.

Pour **ajouter** au fichier (sans écraser l'ancien contenu) :
`ls -1 >> liste.txt`

2>	redirection de la sortie d'erreur
&>	redirection des deux sorties
<	redirection d'entrée, ex. `cat < liste.txt`

Pipes et filtres *103.4*

`ls -1 | wc` sortie de `ls` canalisée vers l'entrée du filtre `wc`.

`find /etc |& wc` StdOut et StdErr fusionnées puis canalisées

Exemples

1. `cat` taper `Ctrl+D` = fin de flux
2. `cat liste.txt | wc -l`
3. `wc -l liste.txt`
4. `wc -l < liste.txt`
5. `cat < liste.txt | wc -l`
6. `wc -l liste.txt l2.txt l3.txt`
7. `cat liste.txt l2.txt l3.txt | wc -l`

Exo. Dessiner le schéma correspondant à chacune des commandes. Identifier filtres et semi-filtres.

Cont Base Fichiers Outils Réseau Scripts Installation Fichiers Filtres FHS Montage

Filtres textes courants *103.2*

Principe Unix : une tâche, un outil.

Beaucoup de filtres fonctionnent ligne par ligne :

- `head` Premières lignes
- `tail` Dernières lignes
- `sort` Trie les lignes
- `uniq` Enlève les doublons
- `grep` Garde les lignes correspondant à une expression donnée. Ex. `ls / | grep v`
- `cut` Conserve les colonnes (resp. champs) donnés
- moins courants : `tr`, `tac`, `paste`, `fmt`...
- paquet `coreutils`

TP : manipulation de texte *103.2*

Le fichier `auteurs.txt` contient une liste d'auteurs avec leur fréquence d'apparition. Ceux qui sont placés entre « ... » sont identifés clairement, à la différence des autres.

1. Séparer énoncé et données dans deux fichiers différents.
2. Combien y a-t-il d'auteurs au total ? Combien de bien identifiés ? De mal identifiés ?
3. Classer les auteurs selon leur fréquence.
4. Lister les 20 auteurs les plus courants, le plus fréquent en premier.
5. Créer un fichier `auteurs2.txt` dans lequels ne figurent pas les auteurs n'ayant qu'une occurrence. Combien sont-ils ?
6. Quels sont les 10 auteurs mal identifiés qui apparaissent le plus souvent ?

TP : synthèse de logs *103.2*

Le fichier `access.log` contient un extrait de logs du serveur Apache, duquel on va essayer de tirer des statistiques.

1. Combien de requêtes sont enregistrées dans le fichier access.log ?
2. Extraire du fichier access.log la liste des adresses IP clientes.
3. Compter le nombre d'occurrences de chaque IP, puis compter le nombre d'adresses IP différentes.
4. Présenter la liste par nombre décroissant d'occurrences.
5. Afficher uniquement les IP ayant effectué au moins 10 accès.
6. Question subsidiaire : pour chacune des IP de la liste précédente, effectuer une résolution de nom (commande host).
   a en passant par un fichier temporaire
   b sans intermédiaire, en une seule ligne de commande

Pour aller plus loin : Sed et Awk 103.2

- Sed : Stream Editor
  - adapté aux opérations sur les chaînes et les regexp
  - `sed -e "s/Old/New/g" f-in > f-out`
- AWK : un langage-filtre
  - pour les fichiers structurés en colonnes ou en champs
  - `gawk -F: '$3 > 999' /etc/passwd`
- Encore plus loin : Perl

Filtres - utilisation avancée 103.2

- La commande `tee` : brancher une dérivation
  `egrep ":[0-9]:" /etc/passwd | tee listing | wc -l`
- La commande `xargs` : transformer STDIN en arguments
  ex. `find /etc/ -size +100k | xargs wc -l`
- La commande `mkfifo` : créer un pipe nommé

```
mkfifo listing
cut -d: -f1-3 listing
egrep ":[0-9]:" /etc/passwd | tee listing | wc -l
```

  ⟶ synchronisation forcée des processus

Fonctions avancées du shell

Quelques techniques non présentées ici :

- autocompletion
- raccourcis clavier
- historique
- fonctions et alias
- boucles, variables et opérateurs
  Permet de réaliser des scripts shell.

Fonctionnement configurable et propre au shell utilisé : sh, bash, tcsh, ksh, zsh...

Différences avec Windows

- Pas de notion de lecteur C: D: etc.
- Tout est dans une même arborescence, de racine /
- Les répertoires sont séparés par des / et non des \
- Existence de liens symboliques
  `ln -s fichier lien`
  Sous windows, les liens sont de simples fichiers *.link*
- Des permissions explicites

Principaux types de fichiers

- Trois principales distinctions :
  - texte ou binaire
  - exécutable ou pas
  - installé par la distribution ou pas
- Quelques exemples :
  - programmes binaires, ex. `/bin/cp`
  - scripts shell, ex. `/etc/init.d/rc.local`
  - fichiers de configuration, ex. `/etc/fstab`
  - fichiers de log, ex. `/var/log/messages`
  - bibliothèques dynamiques **.so**
- Commandes utiles
  - `file` : le type du fichier
  - `which` ou `type` : pour une commande
  - `cat, head, tail` : le contenu du fichier (texte)
  - `hd, ldd, strings...` : le contenu du fichier (binaire)

Cont Base Fichiers Outils Réseau Scripts Installation Fichiers Filtres FHS Montage

Exécutables interprétés et compilés *104.7*

- Langages interprétés
  - interpréteur standard : shell (`bash` ou ...)
  - autres : perl, python, ruby, php
  - *shebang* (ou *hashbang*) : `#! /usr/bin/perl -w`
  - interpréteur nécessaire pour l'exécution
  - code source = exécutable
- Langages compilés
  - entrée : code source texte ex. C, C++, Fortran...
  - chaîne de compilation : `gcc`, `as`, `ld`
  - sortie : binaire exécutable ELF (...)
  - source (C...) ⟶ [compilateur] ⟶ exécutable ELF
  - code source ≠ exécutable

ELF : Executable and Linkable Format *104.7*

Le format standard des exécutables Linux

- Buts
  - Assembler les unités de compilation (*.o)
  - Créer une image mémoire d'un programme
- Trois sous-types de fichiers ELF

EXEC	binaire exécutable
DYN	fichier objet partagé `*.so`
REL	fichier relocalisable `*.o, *.a`

- Commandes disponibles
  - `file /bin/ls` ⟶ ELF 32-bit LSB executable [...]
  - Pour aller plus loin : `readelf -h`, `nm`, `objdump`

Pour aller plus loin : file et MIME

Comment déterminer un type de fichiers ?

- Plusieurs concepts à distinguer
  - l'extension du fichier (si elle existe) : métadonnée
  - sa signature (si elle existe)
  - son type MIME (Multipurpose Internet Mail Extensions)
  - les applications le prenant en charge
- Techniquement
  - `libmagic` à la base de `file` : `man magic`
  - `file -i` renvoie le type MIME
  - `/etc/mime.types`
  - `/etc/mime-magic` et `/etc/magic`

Filesystem Hierarchy Standard 1/2 *104.7*

Norme FHS maintenue par la Linux Foundation

/	racine
/bin/	exécutables principaux (système)
/sbin/	exécutables d'administration (superuser)
/etc/	configuration du système
/home/	répertoires utilisateurs
/root/	homedir de root
/usr/	programmes (gérés par la distribution)
/usr/bin/	exécutables des programmes
...	
/usr/local/	programmes (hors distribution)
/var/	données variables
/var/log	fichiers de log
/var/spool	fichiers tampons (mail, impressions. . .)

Filesystem Hierarchy Standard 2/2 *104.7*

Extensions...

/opt	applications installées hors conventions Unix
/mnt	montages externes (réseau...)
/media	montages amovibles (CD, clé USB...)
/srv	données utilisées par les services (FTP, WWW...)
/selinux	réservé pour Security Enhanced Linux
/run	données runtime (remplace /var/lock et /var/run) (non-LSB)

Systèmes "virtuels" (tout est fichier...)

/dev	fichiers-périphériques
/proc	informations sur les processus : `man 5 proc`
/sys	informations système

Points de montage *104.7*

Comment accéder à un CD-ROM sans D: ?
`mount /media/cdrom`

Les points de montage

Initialement, seule existe la racine /.
Puis `mount` sert à associer

- un périphérique physique (disque, partition) ex. `/dev/sda2`
- un répertoire ex. `/mnt/windowsC`

Exemple : `mount -t vfat /dev/sda2 /mnt/windowsC`

Les montages par défaut sont décrits dans `/etc/fstab`.

`mount` (sans argument) liste les montages en cours.

Pour aller plus loin

- automontage : clés USB, périphériques *hotplug*
- montage par l'interface graphique

Gestion des fichiers

Processus et tâches *103.5*

Gestion des tâches (jobs)

- `commande &` : lancer en arrière-plan
- `jobs (-l)`
- **Ctrl-Z** : met en pause
- **Ctrl-C** : arrête
- `bg` : redémarre en arrière-plan le processus en pause
- `fg` : remet en avant-plan

Affichage des processus

- `top` : affiche les ressources consommées par les processus
- `ps` : Process Show
- `pstree` : **arbre** des processus ⟶ init
- `prtstat` : (paquet `psmisc`) tous les détails d'un processus
- `qps` : interface graphique conviviale

Commande ps - les options *103.5*

1. syntaxe BSD : `ps U root`, `ps aux`
2. syntaxe SysV : `ps -U root`, `ps -ef`
3. syntaxe longue GNU : `ps --user root`

Principales options

1. Options de sélection
   - -e, -A : tous les processus
   - -C <liste commandes>
   - -G, -U . . . <liste utilisateurs, groupes>
   - -t, --tty <liste de terminaux>
2. Options de niveau de détail
   - -f, -F : full, extra-full
   - -o, -O, --format : personnalisé Ex. `ps -O ppid,pgrp,sess`
3. Options d'affichage
   - --sort : tri Ex. `--sort tt,-pid`
   - -H, --forest : hiérarchie
   - --headers --lines=20 : répéter l'en-tête toutes les 20 lignes

Regroupement de processus 1/2 - sessions *103.5*

- **session (SESS)** : processus liés à un même terminal (TTY)
- *session leader* (bash...) : fournit son PID à la session

Terminaux et pseudo-terminaux

- Consoles virtuelles (TTY)
  - consoles texte standard (Alt + F1-F6...)
  - `/dev/tty1-63`
- Pseudo-terminaux (PTYs)
  - terminaux X, session shell...
  - `/dev/pts/0...` + `/dev/ptmx` (System V)

Cont Base Fichiers Outils Réseau Scripts Installation Procs Ordonnancement Signaux... Editeurs Courant

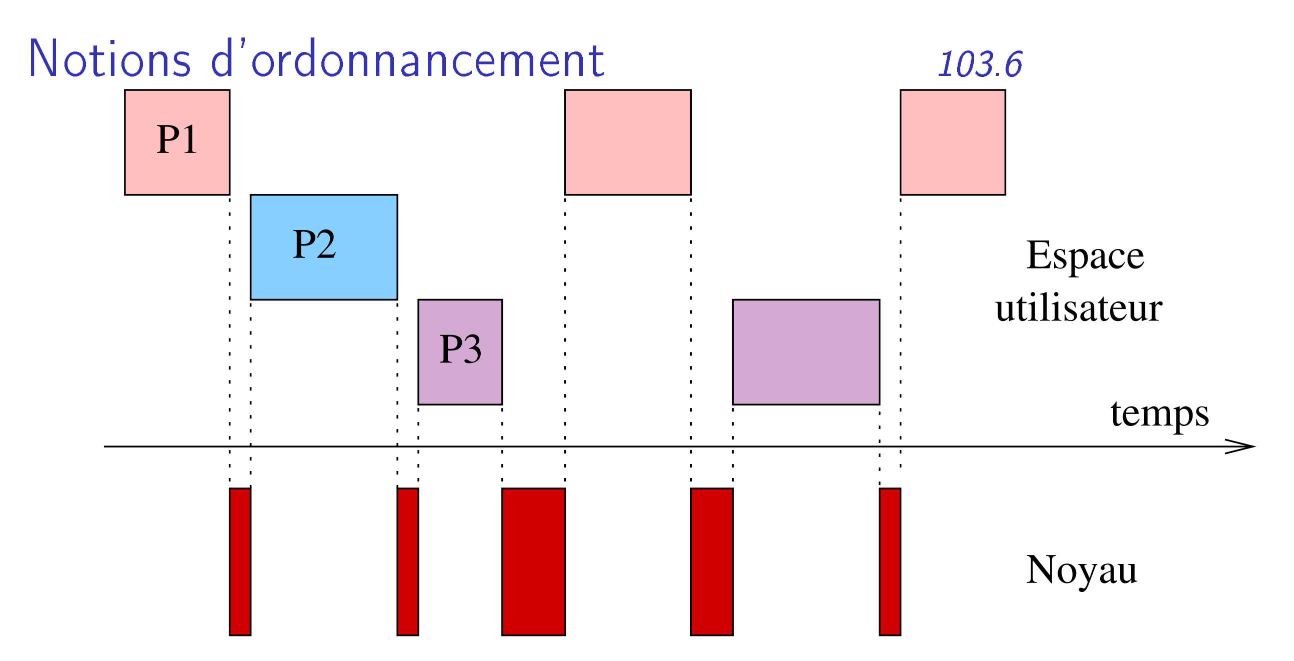
Regroupement de processus 2/2 - groupes *103.5*

Regroupements de processus

- **groupe (PGRP)** : processus formant une même commande (=job)
  ex. `find / | grep pass | less`
- **TPGID** : groupe au premier plan du terminal du processus

Notions d'ordonnancement

103.6



Paramètres

- fréquence : réactivité du système
- proportion : priorités des processus (cf *niceness*)
- charge : nb moyen de processus dans la file (`top`, `uptime`)

Cont Base Fichiers **Outils** Réseau Scripts Installation Procs **Ordonnancement** Signaux... Editeurs Courant

Processus - états et ordonnancement

103.6

Etats des processus

R	demande d'exécution (Running)
S	attente interruptible (Sleep)
D	attente non interruptible
T	stoppé (par SIGSTOP)
Z	"zombie" (ou defunct)

<	priorité haute
N	priorité basse (Nice)
s	session leader
l	multi-thread
+	groupe d'avant-plan

Trois classes d'ordonnanceur (CLS)

- TS : Time Shared (standard)
- FF : Real Time Fifo
- RR : Real Time Round Robin

Priorité et “courtoisie” *103.6*

Courtoisie (*niceness* NI)

- un nombre entier, entre -20 et 19
- -20 à -1 : réservé à root, priorités hautes
- 0 : valeur par défaut
- 1 à 19 : accessibles à tous, priorités basses

Priorité (PRI) : calculée à partir de la courtoisie

- $PRI = 19 - NI$ en temps partagé
- $PRI = 19 - NI + 100$ en temps réel (FF, RR)

Commandes

- nice commande *Ex : nice -n10 md5sum cd.iso*
- renice courtoisie PID *Ex : renice +20 5124*

Cont Base Fichiers Outils Réseau Scripts Installation Procs Ordonnancement Signaux... Editeurs Courant

Contrôle des processus et signaux *103.5*

Rechercher un processus

`pgrep` : recherche multicritères

Arrêter un processus

- `kill` [options] PID `kill -TERM 1955`
- `killall` commande `killall gimp`
- `pkill` [-signal]

Les principaux signaux

- SIGTERM (15) : terminer normalement (“proprement”)
- SIGKILL (9) : terminaison forcée (non ignoré)
- SIGSTOP (19) : arrêt temporaire (pause) (non ignoré)
- SIGCONT (18) : reprise d’un processus arrêté
- SIGINT (2) : terminaison interactive (Ctrl-C)
- SIGTSTP (20) : arrêt temporaire interactif (Ctrl-Z)

Pour aller plus loin : threads noyau *103.5+*

Les threads noyau

- le démon `kthreadd (PID=2)`
- et tous ses fils : `ps -f --ppid=2`
- parfois liés à un processeur : `[ksoftirqd/0]`

En pratique

- Combien de threads noyau sont en cours d'exécution ?
- Quel est le premier "vrai" processus utilisateur ? (hors init)

Processus légers (threads) *103.5+*

Les threads : des "sous-processus"

Partage de : **code**, données, E/S fichiers, signaux, *pile*

Les threads utilisateurs : affichage avec `ps`

- `ps -L -f` : LWP (pid du thread), NLWP (nombre de threads)
- `ps -Lf -m` : sous-processus affichés après les processus "principaux"

En pratique

- Combien de processus multi-threadés tournent ?
- Combien de threads au maximum ? Pour quel processus ?

TP – Processus *103.5*

1. Combien, approximativement, de processus ont été créés depuis le dernier démarrage du système ?
2. Lister les processus **bash** en cours.
3. Utiliser `top` pour trouver le processus utilisant le plus de mémoire. Tenter de l'arrêter.
4. Faire le lien entre `/proc/` et les processus. Cf `man 5 proc`
5. Trouver le processus de PID maximal, puis le dernier processus lancé
6. Chercher le taux de création des processus (en p/s).
7. Créer une fonction pour rechercher le père d'un processus donné, puis une autre pour déterminer la profondeur d'un processus donné (en argument)

Pour aller plus loin...

- Surveiller un processus avec `watch`
  `watch ls -l /var/log/messages`
  `watch -d ps -F`
- Utiliser `wait` (interne) pour synchroniser des tâches (script)
- Utiliser `procinfo`
- Utiliser `unhide` pour chercher les processus dissimulés (rootkits...)
- Utiliser `pidstat` pour obtenir les ressources utilisées (paquet `sysstat`)

Éditeurs de texte 103.8 ?

Éditeurs sans interface graphique

- parfois nécessaire (connexion réseau, problème graphique)
- plus rapide

- nano
  - simple d'utilisation
  - installé par défaut
- emacs -nw
  - puissant et configurable
  - généralement utilisé en mode graphique
- vi / vim
  - éditeur modal : déroutant au premier abord
  - puissant et efficace pour l'administration système

Cont Base Fichiers Outils Réseau Scripts Installation Procs Ordonnancement Signaux... Editeurs Courant

vi / vim 103.8

- Historique Vi
  - `qed` → `ed` (K. Thomson) → `ex` → `vi`
  - 1976 par Bill Joy, étudiant à Berkeley (puis `csh`, `NFS`, Sun)
  - mode *visuel* de `ex` : premier éditeur "pleine page"
  - POSIX (IEEE 1003.2, Part 2 : Shell and utilities)
  - Développement stoppé en 1985 (licence Sun)

- Nombreuses variantes
  - `elvis`, Steve Kirkendall (Minix, Slackware), 1990-2003 ?
  - `nvi`, Keith Bostic (4.4BSD et dérivés libres), 1992-1996 ?
  - `vile` : VI Like Emacs

- VIM (Vi IMproved)
  - auteur Bram Moolenar (NE)
  - 1991 (1.0) - 2008 (7.2)...
  - toutes plateformes : Unix, Linux, Windows...
  - interfaces graphiques : gtk et gnome

vim - en pratique *103.8*

Fonctionne par modes : commande, édition, visualisation.

Raccourcis principaux

Esc	sortir du mode courant
i	insérer (insert)
yy	copier une ligne (yank)
dd	coupe une ligne (delete)
p	coller (put)
:w	écrire dans le fichier (write)
:q	quitter vim (quit)

Pour aller plus loin

- 5dw $\longrightarrow$ efface 5 mots
- yf, $\longrightarrow$ copie le texte jusqu'à la prochaine virgule
- vimtutor pour s'entraîner aux manipulations

vim - Fichiers de configuration *103.8*

Fichiers de configuration

- `/etc/vim/vimrc` : global système
- `/.vimrc` : personnel, ex. :

```
syntax on
set nu
```

Fichiers auxiliaires

- `/.viminfo` : historique commandes, tampons ...

Emacs -

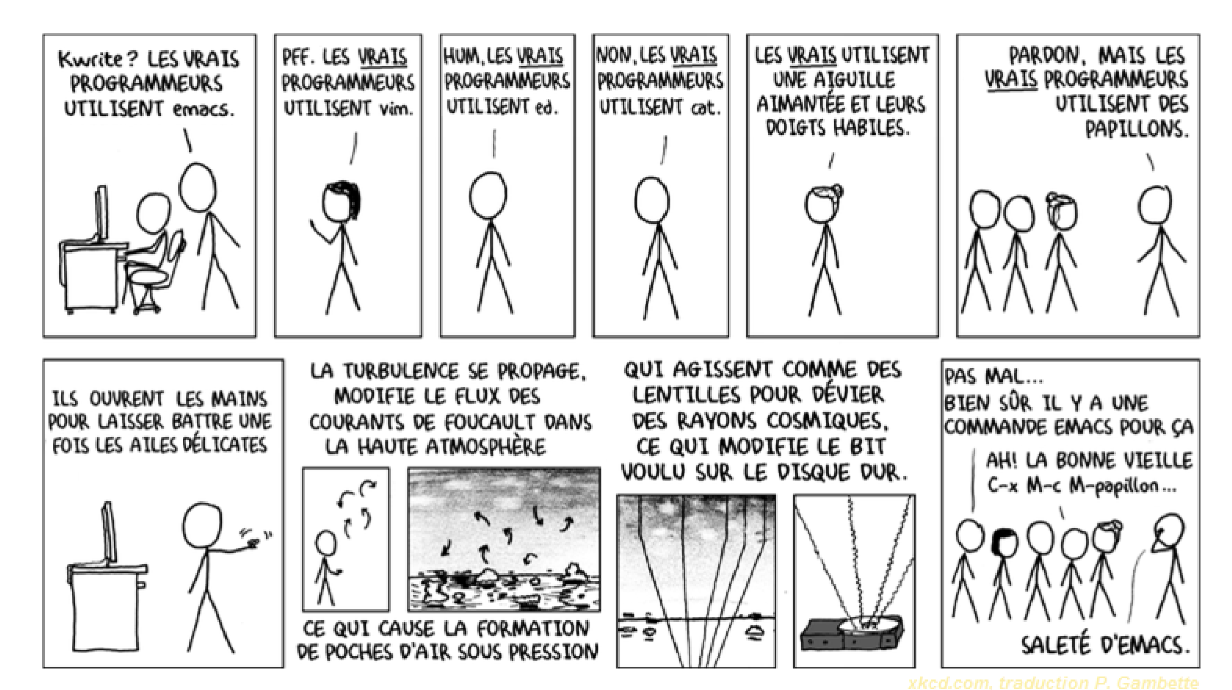
- Historique
  - 1976 : TecoEmacs, Steele et Stallman (MIT) sur PDP/ITS
  - 1978 : MulticsEmacs (B. Greensberg), Lisp langage d'extension
  - 1981 : GoslingEmacs (J. Gosling), 1ère version Unix
  - 1984 : intégré au projet GNU, réécrit (R. Stallman)
  - 1985 : **GnuEmacs 15.34**, 1ère version largement diffusée

  ...
  - juin 2007 : GNU Emacs 22.1
  - sep. 2008 : GNU Emacs 22.3
  - Emacs 23 en préparation (Unicode natif)
  - voir `http://www.jwz.org/doc/emacs-timeline.html`
- Variantes
  - XEmacs (1991-) Lucid Inc.
  - MicroEmacs, plus compact
  - ...

Emacs - en pratique -

- Fonctionnalités
  - fonctionnement "moderne" (monomode)
  - implémenté en langage C
  - extensions en Emacs Lisp (eLisp)
- Trois modes de configuration
  - extension *Customize* (menus, GUI)
  - enregistrement de macros
  - utilisation d'eLisp et fichier `.emacs` ou `.emacs.d/*`

XKCD 378



http://www.lirmm.fr/~gambette/xkcd/index.php?id=378

Compilation d'un exécutable

- Exemple : compilation de `ncdu`
- Procédure standardisée : utilisation d'autoconf/automake
  - `./configure (--help)`
  - `make`
  - `make install`
- Dépannage : recherche de dépendances (bibliothèques dynamiques)

Diff et Patch

- Commande `diff`
  - direct : entre deux fichiers
  - -c, -u : contexte, unifié
  - -r : récursif (entre répertoires)
- Commande `patch`
  - syntaxe `patch -p0 <patchfile`

Réseau utilisateur

Architecture TCP/IP *109.2*

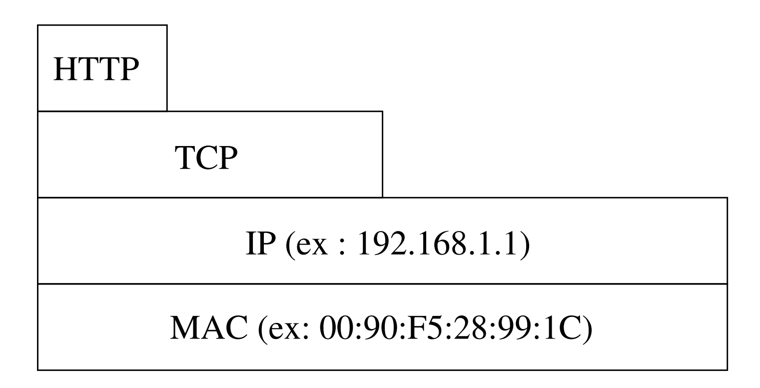
Un modèle par couches

Ethernet réseau local Ethernet-MAC

IP l'adressage Internet, avec une double fonction

- identifiant unique de l'hôte sur le réseau (*identifier*)
- emplacement sur le réseau (topologie) (*locator*)

TCP le transport



Commandes de diagnostic *109.2*

ifconfig

- lo *(interface virtuelle boucle locale)*
- eth0 *(première interface ethernet)*

- l'adresse MAC : 6 octets
  ex. *HWaddr : 00 :90 :F5 :28 :99 :1C*
  Propre à la carte réseau
- l'adresse IPv4 : 4 octets, 32 bits
  ex. *inet addr : 192.168.1.1*
- l'adresse IPv6 : 16 octets, 128 bits
  ex. *inet6 : fe80 : :219 :66ff :fee9 :381/64*

Commandes de diagnostic - 2 *109.2*

- `ping` Tester soi-même, un voisin, un absent, le réseau...
  - `ping -a -c5 192.168.1.1`
  - `ping -b 192.168.1.0`
- `traceroute` (champ TTL)
  affiche le chemin suivi par un paquet (tous les routeurs)
- `mtr` (my traceroute)
  combinaison des deux précédentes commandes

Résolution de noms (DNS) *109.4*

En local : /etc/hosts

Établit des correspondances *nom d'hôte* ⇆ *adresse IP*

Domaine Name Server (DNS)

- Permet une équivalence entre nom et adresse IP.
  Ex : coriolan.silecs.info ⇆ 82.233.121.16
- Fonctionnement par hiérarchie de serveurs

Clients DNS

- Client léger : `nslookup`
- Clients complets :
  - `dig` (dnsutils)
  - `host` (host)
- Sans oublier `ping` (/etc/hosts puis DNS)

Exemple de service : SSH

SSH : connexions sécurisées *110.3*

La famille SSH

- `sshd` : le serveur
- Les clients essentiels
  - `ssh`, `slogin` : connexion interactive ou batch
  - `scp` : copie de fichiers via ssh
  - `sftp` : émulation ftp via ssh
- Les utilitaires
  - gérer les clés utilisateurs : `ssh-keygen`, `ssh-copy-id`
  - mémorisation des clés : `ssh-agent`, `ssh-add`

Remarques

- conçu pour remplacer `rlogin`, `rcp`...
- X11 forwarding : ouverture à distance d'applis graphiques

Utilisation de base des clients SSH *110.3*

- Shell interactif `slogin`
  - `slogin user@distant`
  - Variables d'environnement : `env | grep SSH` : `SSH_CLIENT`, `SSH_TTY`, `SSH_CONNECTIONS`
  - Qui est là ? commandes `who` et `w -l`
  - `slogin -X user@distant` *X11 forwarding*
- Copie distante `scp`
  - `scp user@distant:/home/user/.bashrc ./bashrc` *pull*
  - `scp ./fichier.txt user@distant: /Linux/` *push*
- Shell non-interactif (commande à distance) `ssh`
  - `ssh user@distant /bin/ls`
  - `ssh user@distant "cat /etc/passwd | grep /home"`
  - `ssh user@distant "cat /etc/passwd" | grep /home`
- TP pour aller plus loin : copie réseau en flux avec `tar` et `ssh`.

Cryptographie symétrique et asymétrique *110.3*

Chiffrement symétrique

Une seule clé pour le chiffrage et le déchiffrage

Chiffrement asymétrique

- Principe
  - une clé privée + une clé publique
  - une clé chiffre, l'autre déchiffre
  - secret : chiffrement avec la clé publique du destinataire
  - authentification : chiffrement avec la clé privée de l'expéditeur
  - une infrastructure de distribution des clés publiques (PKI)
- Diversité des clés SSH
  - clés d'hôtes (systématiques) vs clés d'utilisateur (optionnelles)
  - clés RSA vs DSA : deux algorithmes différents
  - clé publique vs privée

Authentification utilisateur SSH par biclé *110.3*

1. Création de la clé
   `local> ssh-keygen -t rsa -C "commentaire" [-f ~/.ssh/ma-clef]`

2. Installation de la clé publique
   `local> ssh-copy-id [-i ma-clef] [user@]distant`
   ou bien `scp` + `slogin` + `cat ... >> authorized_keys`

3. Connexion sans mot de passe
   `local> slogin [-i ~/.ssh/ma-clef] user@distant`

4. Pour aller plus loin : TP utilisation d'un agent SSH
   4.1 Recréer une clé protégée par un mot de passe
   4.2 Comment ne pas retaper le mot de passe ?
   4.3 `ssh-agent` cf gnome-keyring...
   4.4 `ssh-add  /.ssh/ma-clef` puis `ssh-add -l`

Cont Base Fichiers Outils **Réseau** Scripts Installation IP SSH

Complément : configuration SSH *110.3*

Exemple de fichier `/home/USER/.ssh/config`

```
Host eniac
Hostname eniac.moore.upenn.edu.
IdentityFile /home/gallegre/.ssh/eniac_rsa
User gallegre
Port 22

Host hal
Hostname hal9000.nasa.gov.
ServerAliveInterval 30
ServerAliveCountMax 120
```

`man 5 ssh_config`

Shells et scripts

Panorama des shells - 1/2 *103.1*

- Référence
  cf Wikipedia, *Comparison of command shells*
- Shells historiques
  - `sh` original (1971), K. Thompson, Unix AT&T
    mode interactif seulement
  - Bourne shell (`sh`, 1977), Bell Labs, Unix v.7
    ajout des scripts
  - C shell (`csh`, 1978), Bill Joy, Unix BSD
    descendant du Thompson, syntaxe plus proche du C

Panorama des shells - 2/2

103.1

- Shells courants
  - tcsh (1981, Tenex C shell), Ken Greer (Carnegie-Melon U.) par défaut sur FreeBSD
  - ksh (1982), Korn shell, Bell Labs : longtemps propriétaire évolutions ksh88 (POSIX), ksh93
  - bash (1987) Bourne Again Shell (projet GNU) par défaut sur GNU/Linux (GPL) ; v4.0 en février 2009
  - zsh (1990), Paul Falstad (Princeton U.) probablement le plus riche en fonctionnalités
- Shells restreints
  - (d)ash, Kenneth Almquist (sh compact)
  - sash, stand-alone shell (commandes internalisées)
- Changer de shell par défaut : chsh

Les fonctionnalités du shell

103.1

- Mode interactif
  - complétion automatique
  - historique des commandes, recherche... (readline)
  - alias
  - ...
- Fonctionnalités mixtes
  - boucles (for, while...)
  - enchaînements de commandes et valeurs de retour
  - fonctions
  - développement (globbing, variables...)
  - fichiers de configuration (bashrc...)
  - ...
- Mode script
  - gestion des paramètres ($1, $2...)
  - tests et conditions (if ... then ... else)
  - ...

Documentation 103.1

- Documentation électronique
  - `man bash`
  - `help help`
- Documentation libre
  - *Advanced Bash Scripting Guide*, M. Cooper (6.0, mars 2009)
    VF : *Guide avancé d'écriture des scripts Bash* (5.3)
  - *Bash Guide for Beginners*, M. Garrels (1.11, déc. 2008)
    VF : *Guide Bash du débutant* (avril 2007)
  - nombreux *tutoriels bash*, plus courts ou plus ciblés
- Livres
  - *Programmation shell sous Unix/Linux*, Ch. Deffaix Rémy, ENI
  - *Introduction aux scripts shell*, A. Robins, N. Beebe, O'Reilly

Complétion 105.1

Complétion standard

- noms de commandes
- entrées de répertoires (fichiers...)

Complétion étendue

- `shopt -s progcomp`
- `source /etc/bash_completion`
- sous-commandes
- options longues
- fichiers distants (ssh...)
- ...

Readline - historique *105.1*

- history
  stockage dans /.bash_history
- édition accélérée
  - C-a, C-e, C-←, C-→ : déplacements
- recherche et parcours de l'historique
  - man readline + /etc/inputrc : fichier de configuration
- développement de l'historique
  - indicateur d'événement : ex. !!, !123, !#
  - indicateur de mots : ex. 0, 1, ^, $
  - modificateurs : ex. ^chaine1^chaine2^

Les alias *105.1*

Quelques exemples

- alias ls="ls -color=auto"
- alias ll='ls -l'
- alias today='date +"%Y%m%d"'
- alias rm='rm -I'

- alias — seul : liste les alias définis
- unalias (-a) — détruit un alias défini

Pour aller plus loin : les fonctions

utilisation interactive : “alias à arguments”
ex. function prefix { mv $1 $2$1 ;}

Fichiers de configuration *105.1*

- Fichiers principaux
  - `/home/USER/.bash_profile` : shells de login
  - `/home/USER/.bashrc` : autres shells
  - `/etc/profile`
  - `/etc/bash.bashrc`

- Contenu
  - Variables d'environnement
    p.ex. prompt : `$PS1, $PS2...`
  - alias
  - fonctions
  - réglages du shell
  - inclusions (`source`)

Configuration du shell *105.1*

- Variables d'environnement
  - `$SHELLOPTS`
  - `$PS1`, `$PS2` ...
  - `$GLOBIGNORE` ...

- `set -f/+f` ou `set -o OPTION`
- `help set`

- `shopt -s / -u` (set / unset)
  - env. 40 options booléennes : `shopt -p`
  - + 27 options "à la set" : `shopt -o -p`

Bash - les “développements” *105.1*

Sept types de développements successifs (*expansions*)

1. développement des accolades { } : factorisation
2. développement du tilde ~ ou ~user
3. développement des paramètres et variables
4. substitution de commande : `‘cmd‘` ou `$(cmd)`
5. développement arithmétique
6. découpage en mots
7. développement des chemins (globbing)

Rappel : les protections

- guillemets doubles
- guillemets simples : plus “forts”
- antislah : un caractère

Diagnostic et enchaînements *105.2*

Valeurs de retour et booléens du shell

- `$?` : valeur de retour du dernier processus terminé
- 0 = OK $\Longrightarrow$ vrai !
- >0 = erreur $\Longrightarrow$ faux !

Enchaînement des commandes

- ET : `mkdir Toto && cd Toto`
- OU : `mkdir Titi || echo "erreur d’écriture"`
  → `mkdir Tutu && echo "OK" || echo "impossible"`
- enchaînement : `cmd1 ; cmd2`
- en parallèle + arrière-plan : `cmd1 & cmd2` $\&\& \neq \&$

Les fonctions

105.1

La commande function

```
function lprman
    {
    man -t $1 > $1.ps
    lpr $1.ps
    }
```


Métaprogrammation

103.4

- La commande `xargs`
  ex. `find /etc/ -size +100k | xargs wc -l`
- La substitution de commande
  ex. `wc -l $( find /etc -size +100k )`
  ou `` wc -l `find /etc -size +100k` `` (backquotes)
  `echo "Vous êtes connecté sur $(uname -n)."`
- Remarque : la substitution de commande est plus générique (mais plus gourmande).

Redirections étendues : HERE... *105.1*

- HERE-Documents <<

```
$ wall <<FIN
> ETEIGNEZ VOS MACHINES
> coupure electrique imminente
> --- l'equipe systeme
> FIN
```

- HERE-Strings <<<
  ex. `cut -b cut -b 1,3-5,16- <<< "internationalisation"`

Développement des paramètres et variables *105.1*

Évaluation arithmétique *105.1*

- Bash standard

```
i=0
i=$i+1      # "0+1"
i=i+1       # "i+1"
```

- Typage numérique (entier)

```
declare -i n=5
n=n+1     # "6"
a=n+1     # "n+1"
```

- Évaluation arithmétique

```
i=0
i=$(( i+1 ))     # standard
(( i=$i+1 ))     # extensions bash...
(( i=i+1 ))
let i=i+1
let i=$i+1
```


Écrire une boucle numérique *105.2*

- La commande `seq`
  - `for i in $(seq 0 2 8) ; do echo $i ; done`
  - `seq 8` — 1 à 8
  - `seq 0 8` — 0 à 8
  - `seq 0 2 8` — 0, 2, 4, 6, 8

- Bash, mode standard
  `while [ $i -lt 9 ] ; do echo $i ; let i=i+1 ; done`

- Bash, mode arithmétique
  1. `while ((i<9)) ; do echo $i ; done`
  2. `for ((i=0; i<9; i+=2)); do echo $i; done`

- Bonus : formatage numérique
  1. `printf 'James Bond %03d, No %02d' 7 3`
  2. `seq -f '%03.0f' 0 2 12` — format virgule flottante (!)

Scripts shells 105.2

- Modèles d'exécution
  - exécution `bash monscript.sh`
  - ou exécution avec `# /bin/sh`
  - inclusion : `source script`
- Paramètres positionnels
  - $0, $1, $2...
  - $# nombre d'arguments
  - "$*" la liste des arguments, sans tenir compte des blancs
  - "$@" la liste des arguments, en tenant compte des blancs
  - shift

Un exemple : bonjour.sh 105.2

```
#! /bin/sh

echo "je suis $$"
echo "bonjour $NAME"
NAME="Guillaume"
echo "bonjour $NAME"
exit 0
```

Rappel

$$: numéro du processus courant

Panorama des structures de contrôle *105.2*

- Tests
  - `test` ou `[ ... ]` — test standard
  - `[[ ... ]]` — test avancé (Bash)

- Conditions
  - `if ... then ... fi`
  - `if ... then ... elif ... else ... fi`
  - `case MOT in MOTIF) ... esac`

- Boucles
  - `for VAR in VALEURS ...; do ... done` — énumération
  - `for (( E1; E2; E3 )) ; do ... done` — numérique
  - `while ...; do ...; done` — tant-que
  - `until ...; do ...; done` — until
  - `select MOT in VALS ; do ... done` — menu (boucle interactive)

Exemples de tests *105.2*

- Tests standard `[ ... ]`- exemples

- Tests avancés `[[ ... ]]`- exemples

Boucle for *105.2*

- Usage interactif (ligne de commande)
  - `for VAR in un deux trois ; do echo $VAR ; done`
  - `for F in *.txt ; do wc -l $F ; done`
- Usage en script

```
for ARG in $@
    do
        echo $ARG
        ...
    done
```

- Variante `select` (en script)

while et until *105.2*

La condition `if` *105.2*

- En ligne de commande
  - `if mkdir Rep ; then echo Fait ; else echo Erreur ; fi`
  - cf `mkdir Rep && echo "Fait" || echo "Erreur"`

La comparaison `case` *105.2*

105.2

TP scripts 1 : disable/enable

105.2

1. Ecrire un script `disable.sh` qui
   - prend en argument un nom de fichier
   - le renomme en lui ajoutant le suffixe `.OFF`
2. Ecrire le script inverse, `enable.sh`, qui supprime le suffixe `.OFF`. Il doit accepter en argument les deux variantes `fichier` et `fichier.OFF`.
3. Transformer les deux scripts en un seul `xable.sh`, qui prend une option (-d ou -e) pour indiquer le sens de l'opération.

TP scripts 2 - boucles *105.2*

- Avec `find`
  - Exo : Trouver le nb d'entrées de répertoire de chaque type sous `/`, sans changer de système de fichiers (`-xdev`).
  - Exo : Pour les quatre types minoritaires, afficher les entrées.

TP scripts : gestion des liens *105.2*

1. Ecrire un script `rmlink.sh` qui
   - prend en argument une entrée de répertoire
   - la supprime si c'est un lien symbolique
   - retourne un message d'erreur sinon
2. Variante `rmbrlink.sh` : supprime seulement les liens cassés
3. Variante : transforme `rmbrlink.sh` en option (-b) de `rmlink.sh`
4. Ecrire un script `rmhlink.sh` qui supprime l'entrée de répertoire si c'est un fichier régulier avec (ref>1), autrement dit si c'est un lien dur.
5. Ecrire une fonction `ireadlink` qui affiche une résolution de lien symbolique avec intermédiaires : ex. `/usr/bin/rsh -> /etc/alternatives/rsh -> /usr/bin/ssh`.

Tableaux en Bash 1/2 : index numériques *105.2*

- Exemples

```
declare -a Tab

Tab[0]=zero
Tab=(zero un deux trois quatre cinq)

echo ${Tab[2]}
echo ${Tab[*]}
echo ${Tab[*]:2:3}
```

- TP

Cont Base Fichiers Outils Réseau Scripts Installation shells interactif mixte scripts sed... awk

Tableaux en Bash 2/2 : tableaux associatifs *105.2*

- Tableaux associatifs (depuis Bash 4)

```
declare -A Asso
Asso[couleur]=rouge
Asso=([couleur]=rouge [outil]=marteau [animal]=lion)

declare -p Asso
echo ${Asso[couleur]}
echo ${Asso[*]}
echo ${!Asso[*]}
for KEY in ${!Asso[*]}; do
  echo "$KEY => ${Asso[$KEY]}"; done
```

- TP : Trouver la place occupée par les fichiers de chaque type MIME dans le répertoire utilisateur.
  Astuce : utiliser la commande `file -i` pour les types MIME.
  Variante : remplace le type MIME par l'extension.

sed, expressions rationnelles

Expressions rationnelles (ou régulières) *103.7*

- un “outil” commun à de nombreux utilitaires
  `grep, sed, awk, vim`...
- Deux formes (malheureusement !)
  - forme “basique” interne à chaque commande
  - forme “étendue” standardisée (POSIX.2)
  - `man 7 regex`

sed - Stream EDitor *103.7*

Contexte

- écrit par Lee McMahon en 1973/1974 (Bell Labs),
- dérivé de l'éditeur monoligne `ed`
- applique une série de règles d'édition de texte...
- à chaque ligne d'un fichier, successivement
- reconnaît deux types d'expressions régulières

Quelques exemples

- `sed -e "s/Old/New/g" f-in > f-out`
- `sed -e '/^ *$/d' f-in`

Cont Base Fichiers Outils Réseau Scripts Installation shells interactif mixte scripts sed... awk

awk

AWK - un filtre-langage

- Origines...
  - langage défini par Aho, Weinberger, Kernighan en 1977
  - standard POSIX, NAWK (New AWK), courant 1980s
  - *The AWK Programming Language*, 1988
  - plusieurs interpréteurs libres (orig-awk, gawk, mawk...) ou pas
  - une syntaxe intermédiaire entre C et le shell
  - à l'origine de Perl
- Caractéristiques principales
  - conçu pour analyser un fichier (ou flux) texte divisé en champs
  - tableaux associatifs
  - expressions régulières
  - bien adapté à des scripts unilignes (comme `sed`)
- Particularités des implémentations

mawk	performances et efficacité (précompilé)
gawk	richesse et documentation (i18n)
xgawk	extensions XML, PostgreSQL
awka	comilateur AWK -> C

AWK - invocation et structure

- Invocation de awk
  - `awk -f script.awk fichier`
  - `awk 'code AWK' fichier`
  - exécutable commençant par `#! /bin/awk -f`
- Structure d'un script

```
motif  { action }
...
```

  - motif : sélecteur de lignes ou BEGIN ou END
  - action : instruction de type procédural
- Quelques exemples

```
awk 'BEGIN { print "Bonjour !" }'
awk 'length($0) > 60' /etc/passwd
awk 'NR % 2 ==0' /etc/passwd
awk 'BEGIN {FS=":"}  NR % 2==0 {print $1}' /etc/passwd
```

AWK - TP avec find

- Utilisation basique
  Trouver la place occupée par l'ensemble des fichiers de plus de 1 Mo dans le répertoire utilisateur (on peut varier les critères...).

- Utilisation avancée : tableaux associatifs
  Trouver la place occupée par les fichiers de chaque type d'extension (txt, sh, ...) dans le répertoire utilisateur
  Astuce : utiliser la directive `split` pour les extensions.

Installation Debian

Choix d'une distribution : Debian *102.4*

Debian

- distribution communautaire non commerciale
- très grand nombre de paquets officiels
- excellente gestion des dépendances et des mises à jour
- toujours 3 "suites" Debian en cours
  - stable (act. 6.0.1 *Squeeze*, sortie en fév. 2011)
  - testing (act. *Wheezy*)
  - unstable (*Sid*)
- réputation d'être moins "conviviale" que la concurrence

Installateur "technique" et flexible.

Installation Debian *102.4*

Plusieurs supports possibles

- CD-ROM / DVD-ROM
- CD-ROM puis réseau
- Clé USB puis réseau
- Tout-réseau : serveur TFTP (PXE / Etherboot)

Installations automatisées

- Préconfiguration (*preseed*)
- FAI (*Fully Automated Installation*) : serveur nécessaire
- Cf Manuel d'Installation, 4.7

Les options de démarrage *102.4*

Plusieurs méthodes

- install (par défaut)
- expert (plus de questions)
- rescue (sauvetage / restauration)
- pour chacune : mode texte ou GUI

Multiples paramètres d'amorçage

- paramètres pour l'installateur Debian
- gestion du matériel problématique : paramètres des modules Linux

Exemple

- `expert-gui locale=fr_FR acpi.blacklist=yes`

Pendant l'installation *102.4*

Phase 1 : installation

- Accès aux consoles : shell, log installateur, log noyau

Phase 2 : redémarrage et configuration

- debconf : base de données des configs de paquets

Cohabitation avec d'autres systèmes *102.4*

boot manager (GRUB/LILO) : choisir le système d'exploitation au démarrage.

Partitions

- Linux peut accéder aux partitions Windows
  L'accès en écriture est par défaut restreint avec NTFS.
- Windows ne peut pas accéder aux systèmes de fichiers Linux.

Gestion des paquets avec APT

Les paquets Debian *102.4*

Paquet binaire (.deb) ou source (.dsc)

Contenu d'un paquet binaire (.deb)

- Archive des fichiers (data)
- Métadonnées (control/control)
  - Descriptions textuelles : courte et longue
  - Section : classement du paquet dans une hiérarchie debian
  - Version
  - Dépendances, conflits, suggestions, recommandations...
  - debtags : indexation du paquet
    Par exemple : network::service, suite::apache
- Utilitaires (control/...)
  - scripts installation / suppression
  - sommes de contrôle (MD5sum)

Travaux pratiques *102.4*

Examen du paquet dpkg

- À la main
  Commandes : `ar t`, `tar -x`
- Avec l'outil dédié
  Commande : `dpkg-deb`

Deux cas particuliers *102.4*

- Méta-paquets
  - Paquet "réel" : le .deb existe
  - Paquet de paquets : via les dépendances
  - Exemple : `gnome`
- Paquets virtuels
  - Paquet virtuel : le .deb n'existe pas
  - Indique un service générique, fourni par plusieurs paquets
  - Exemple : `mail-transport-agent` ; cf `mailman`

dpkg : gestion locale *102.4*

dpkg manipule les paquets debian (`.deb`) sans accès réseau.

Principales options de dpkg

- `dpkg -i paquet.deb` ⟶ installe
- `dpkg -r paquet` ⟶ désinstalle
- `dpkg -L paquet` ⟶ liste les fichiers du paquet
- `dpkg -S fichier` ⟶ recherche `fichier` parmi les paquets installés

dpkg est souvent nécessaire pour les opérations fines (conflits importants, diagnostic, etc.)

TP – dpkg *102.4*

1. Installer ncdu à partir des sources. En quoi est-ce pénible ?
2. Télécharger le navigateur Opera (www.opera.com) et l'installer grâce à dpkg.
3. Avec dpkg, lister les fichiers installés par Opera.
4. Quels exécutables sont fournis par le paquet **sysvinit** ?
5. Quels sont les paquets actuellement installés sur votre machine ?
6. De quel paquet provient la commande ifconfig ?
7. Reconfigurer le serveur mail local.

La famille apt *102.4*

- synaptic
- aptitude
- apt-get
  - update
  - install
  - ...
- apt-cache
  - search
  - show
  - policy

Fichiers

/etc/apt/apt.conf.d/
/etc/apt/sources.list

Documentation

apt-howto-en, apt-howto-fr

TP – apt *102.4*

1. Mettre à jour sa distribution.
2. Examiner le fichier `/etc/apt/sources.list` et en comprendre la syntaxe.
   Quelle est l'organisation d'un miroir Debian ?
3. Ajouter aux sources APT les dépôts de la distribution testing.
   Que se passe-t-il en cas de demande de mise à jour ?
4. Créer `/etc/apt/apt.conf` afin de fixer la version (*release*) par défaut à stable.
   Retenter une mise à jour.
5. **apt** garde une copie de sauvegarde des paquets téléchargés. Effacer ces fichiers.

La dernière évolution : aptitude *102.4*

- Historique
  1. dselect
  2. apt-get
  3. aptitude
- Interfaces
  - Ligne de commande (sous-commandes compatibles `apt-get`)
  - Interface semi-graphique (ncurses)
- Les avancées d'aptitude
  - un log des opérations : `/var/log/aptitude`
  - distinction paquets : installés automatiquement / à la demande
  - résolution des dépendances : meilleure, plusieurs alternatives
- Documentation : `aptitude-doc-en`, `aptitude-doc-fr`

Reconfiguration d'un paquet *102.4*

Debconf

- une mémoire des choix de configuration
- interfaces : dialog, readline, n-i, gnome, kde, (editor, web)
- priorités : low, medium, high, critical
- fichier de configuration : `/etc/apt/apt.conf.d/70debconf`
- base : `/etc/debconf.conf`, `/var/cache/debconf/*`
- manpages : `debconf(7)`, `debconf(1)`, `debconf.conf(5)`

Commandes

- `dpkg-reconfigure <paquet>`
- manpages : `dpkg-reconfigure(8)`, `dpkg-preconfigure(8)`

Le suivi de bugs de Debian *102.4*

BTS : le Bug Tracking System

- `http://www.debian.org/Bugs/`
- intégration à APT : `apt-listbugs`

Déposer un bug

- le paquet `reportbug`